CTSRD Project Briefing

Robert N. M. Watson (Cambridge)
Peter G. Neumann (SRI)       Simon W. Moore (Cambridge)

DARPA CRASH PI Meeting
San Diego, California, USA
14 January 2014
CTSRD at the PI Meeting

Dr Peter G. Neumann
Dr Robert N. M. Watson
Dr Simon W. Moore
Dr Jonathan Anderson
Dr David Chisnall
Dr Nirav Dave
Mr Brooks Davis

Mr Rance DeLong
Dr Khilan Gudka
Dr Theo A. Markettos
Mr Ed Maste
Dr Michael Roe
Mr Colin Rothwell
Mr Stacey Son
CTSRD

- Spans security, CPUs, OS, compilers, languages, program analysis/ transformation, HW/SW formal methods.
- Clean-slate design violates some current conventions, in exchange for dramatic security improvements.
- Capability-based CPU protection and compartmentalization features mitigate known and unknown vulnerability classes.
- Hybrid model facilitates incremental SW adoption.
- Program analysis and transformation techniques improve software TCB correctness, utilize new CPU features.
- Formal methods link models, hardware, and software.
CTSRD project elements

- Capsicum, compartmentalization, and CTSRD
- Capability Hardware Enhanced RISC Instructions (CHERI)
  - CHERI ISA and hardware compartmentalization prototype
  - CHERI platform: tablet, CheriCloud, peripherals, etc.
  - CHERI software: CheriBSD, CHERI Clang/LLVM, apps
  - Architectural extraction, verification of Bluespec (Smten)
  - ISA-level proofs and automated test generation
- Security-Oriented Analysis of Application Programs (SOAAP)
- Temporally Enhanced Security Logic Assertions (TESLA)
August 2013 CTSRD/MRC2 meetings, Cambridge, UK
COMPARTMENTALIZATION
FOUNDATIONS
Application compartmentalization

- Compartmentalization decomposes software into isolated components.
- Each sandbox runs with only the rights required to perform its function.
- This model implements the principle of least privilege.
When a conventional application is compromised, ambient rights are leaked to the attacker, e.g., network and file system access.

Compromising a compartmentalized application yields only held rights to the attacker. As vulnerabilities yield fewer rights, attackers must exploit many vulnerabilities to meet their goals.
Capsicum update

- Hybrid capability model: OS APIs for application compartmentalization
- Joint Cambridge/Google project
- Experimental feature in FreeBSD 9.x; out-of-the-box in 10.0 (RSN)
- FreeBSD Foundation, Google
  - Funded projects will continue in 2014
  - Growing number of FreeBSD programs are using Capsicum out-of-the-box: tcpdump, auditdistd, hastd, etc.
  - Casper framework offers services to sandboxes (e.g., DNS, socket server)
- Google has published a Linux port prototype and hopes to upstream
• Applications can be compartmentalized in different ways, trading off security and performance

• Finer-grained decompositions mitigate vulnerabilities better: attacks yield fewer rights, so attackers must exploit more vulnerabilities to accomplish their goals

• Ideally, web browsers would use hundreds/thousands of sandboxes: one for each image, script, etc.

• However, CPUs support few simultaneous processes; e.g., Google Chrome reuses up to 20 sandboxes, one per tab

• As a consequence of CPU design, malware in a webmail image attachment can access a user’s entire mailbox
Capability hardware enhanced RISC instructions

CHERI PROCESSOR
Capability hardware enhanced RISC instructions (CHERI)

- CHERI hybrid capability model:
  - Fine-grained memory protection
  - In-address-space sandboxing
- Extends 64-bit MIPS ISA
- Haskell-derived Bluespec System Verilog HDL; synthesizes to Altera and Xilinx FPGAs
- Fully pipelined; multithreaded and multicore under development
- Extensive test suite and tools
- ISA and design subject to new formal analysis
- Shortly to be released as open source
**CHERI development timeline**

- **October 2011**: Deimos capability-based microkernel runs first sandbox
- **June 2012**: CheriBSD capability context for user threads
- **July 2012**: LLVM generates Cheri ISA code
- **November 2012**: Sandboxed user code runs on CheriBSD; trojan mitigated
- **December 2012**: Nested sandboxes and sandbox exception handling
- **February 2013**: Cheri box on the Internet
- **January 2014**: CheriBSD now uses Cheri Clang/LLVM for key sandboxing components
- **November 2011**: tPad tablet demo with Deimos
- **May 2012**: DE4 tablet runs FreeBSD
- **April 2013**: CheriCloud online for SRI/Cam users
- **December 2013**: BERI Open Systems CIC created
CHERI hardware platform

- CHERI prototypes
- Tablet prototype: CPU, DRAM, battery, flash, touchscreen, HDMI, Ethernet
- In-field CPU, OS updates
- CheriBSD OS, CHERI SDK
- CHERI demonstrations
  - E.g., fine-grained compartmentalization in CheriPoint presentation package
CheriCloud

- Centralized facility supporting remote CHERI software development for CTSRD and MRC2 projects
- 7 x DE4 FPGA boards in 4U
- CHERI CPU + CheriBSD with Ethernet on each DE4
- Reset and serial console
- SSH into a live CHERI system; off-the-shelf open-source applications
- Total end-user transparency!
CHERI enhancements (CTSRD)

- CHERI ISA v2.1 enhancements to object-capability invocation, software debugging features
- FPU – particularly useful for Olden benchmarks
- Improved hardware ISA-level tracing + CheriVis
- CHERI2 now fully implements ISA v2.1
- Annabella release in July 2013
- Multithreaded CHERI2 boots BSD in simulation
- Multicore CHERI in testing
CHERI enhancements ($(MRC)^2$)

- DARPA MRC sister project also using and enhancing CHERI
- Multithreading and multicore
- Multi-FPGA interconnect
- AXI bus conversion
- NetFPGA 10G
- CPU Tracing enhancements
- FPU maturity
- BlueSwitch
CHERI formal verification: ISA model

- Formal model of ISA described in SAL model checker
- Bluespec CHERI and CHERI2 capability units are automatically tested against the SAL model
- New: we can now automatically translate the SAL model into PVS
- Using PVS, we can prove “memory safety” for a CHERI ISA subset
- Future work: prove security properties of full model
- Future work: prove security properties of key software TCB elements (e.g., CCall, Creturn)
CHERI formal verification: Bluespec analysis / Smten

- Smten: Automatic Translation of High-level Symbolic Computations into SMT Queries
- Motivation: SMT solvers are widely used for model checking, automated theorem proving and test generation, but translating a model into an SMT form is tedious
- Solution: Smten is a high-level, purely functional language, with syntax and features borrowed heavily from Haskell which greatly helps translation of models into SMT queries
- Initial work published at CAV’13
- See Nirav Dave during the poster session for details.
CHERI SOFTWARE
CHERI software model

- Fine-grained userspace memory protection, in-process sandboxing
- MIPS/CHERI binaries tightly integrated (e.g., CHERI library in MIPS binary)
- Compiler allows pointers to be replaced with tagged, bounds-checked capabilities
- In-progress CHERI debugger
- OS support for model, tracing tools, etc.
- Userspace sandbox model, class libraries, components, monitoring tools
- CheriBSD on github so more easily accessible to downstream users
- BERI support + drivers shipping as FreeBSD 10.0 embedded target in weeks
CHERI extensions to FreeBSD

- CHERI register file preserved for each user thread
- CCall/CReturn exception handlers: object-capability invocation
- CHERI “trusted stack” for object-capability return path
- Sandbox fault recovery unwinds trusted stack on MMU fault, capability fault, or other thread traps
- Kernel accepts system calls only from in-process protection domains with ambient authority; requires using the system class
- Kernel debugging extensions for CHERI LLDB
- CHERI memory protection via CHERI Clang/LLVM
- libcheri(3) API to create and invoke sandboxes
- Extensions to the procstat(1) tool to track sandbox state
CHERI Clang/LLVM/LLDB

- CHERI Clang/LLVM
  - Clang supports new qualifiers, builtins for capability manipulation
  - LLVM CHERI code generation extends MIPS support
  - Pointers use MIPS representation and instructions by default
  - Pointers tagged as `__capability` generates CHERI instead of MIPS
  - Experimental CCCured work automatically converts C code to CHERI ISA

- CHERI SDK
  - Complete cross-development environment: toolchain, libraries, headers.

- CHERI LLDB
  - LLDB supports CHERI registers in core files; live debugging in-progress
Example object-capability/sandbox invocation: “hello world”

- Application main() invokes sandbox
- Sandbox invokes system-object puts()
- System object invokes libc puts()
- System-object puts() invokes libc puts()
- libc puts() invokes write() system call
- Kernel performs write()
Example object-capability/sandbox invocation:

Legacy MIPS code can appear throughout the stack, but requires access functions (i.e., copies) to access non-$c0 data.

Rights passed between sandboxes must be described using capabilities.

System-call interface remains largely unmodified: MIPS ISA/ABI.

In the future, we will add hybrid CHERI-aware system calls allowed in sandboxes, but scoped by capability arguments.
libcheri: object-capability sandbox API

- C-language bindings for CHERI object-capability sandboxes
- Sandbox class
  - For now, memory image; soon, ELF binary (or segment)
  - new, method_declare, destroy
- Sandbox object
  - Instantiated class with data
  - new, getsystemobject, cinvoke, destroy
- Small assembly stubs for caller invoke() and callee enter()
libc_cheri: sandboxed C library; libcheri system class

- Subset of key C functions
  - Useful functions usable without ambient authority (e.g., snprintf)
  - Bottom-end functions invoke CHERI system-class object capabilities instead of system calls
- Kernel rejects calls without ambient authority
  - Sandboxes must request operations with ambient effects through CHERI system class
procstat(1): sandbox monitoring

% login -i .ssh/id_cheri_host ctsrd@cheritest.sec.cl.cam.ac.uk
Last login: Sat Nov 16 03:26:50 2013 from ip-64-134-230-112.public.wayport.net
FreeBSD 11.0-CURRENT (CHERI_DE4_SDROOT) #8 825c7e7(master)-dirty: Sat Jan 11 00:35:25 GMT 2014

% procstat -RX 7114

<table>
<thead>
<tr>
<th>PID</th>
<th>COMM</th>
<th>CLASS</th>
<th>METHOD</th>
<th>INVOKE</th>
<th>FAULT</th>
<th>SMIN</th>
<th>SMAX</th>
<th>SMEAN</th>
<th>SMEDIAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>md5</td>
<td>4</td>
<td>0</td>
<td>10116</td>
<td>158925</td>
<td>47478</td>
<td>10436</td>
</tr>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>abort</td>
<td>1</td>
<td>1</td>
<td>3187</td>
<td>3187</td>
<td>3187</td>
<td>3187</td>
</tr>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>helloworld</td>
<td>1</td>
<td>0</td>
<td>452296</td>
<td>452296</td>
<td>452296</td>
<td>452296</td>
</tr>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>puts</td>
<td>1</td>
<td>0</td>
<td>456118</td>
<td>456118</td>
<td>456118</td>
<td>456118</td>
</tr>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>syscall</td>
<td>1</td>
<td>0</td>
<td>6551</td>
<td>6551</td>
<td>6551</td>
<td>6551</td>
</tr>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>divzero</td>
<td>3</td>
<td>3</td>
<td>2900</td>
<td>3166</td>
<td>3005</td>
<td>2950</td>
</tr>
<tr>
<td>7114</td>
<td>cheritest</td>
<td>cheritest-helper.bin</td>
<td>malloc</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Libcheri exports statistics on sandbox classes, objects, and methods
- libprocstat(3) and procstat(1) can query/print this
- libprocstat(3) provided data backend for demo UI
In progress: open sourcing CHERI

- Complete open-source hardware-software research/teaching stack
- BERI Open Systems CIC (“Community Interest Company”) Dec 2013
- BERI Apache-style license (HW), BSD license (SW)
- Physical designs for DE4 tablet, interconnect boards
- CHERI and CHERI2 Bluespec designs; debugging components/tools
- CHERI test suite, formal models
- FreeBSD device drivers
- CheriBSD capability support
- CHERI Clang/LLVM/LLDB
- ETA: January/February 2014
CHERI next steps

• CheriBSD kernel features (e.g., debugging, lazy switching)
• “Pure” CHERI ISA support for Clang/LLVM
• CHERI LLDB full feature support
• CCured-like automated use of memory protection
• Further CHERI ISA refinements: e.g., explicit CNULL
• Shift stack, heap access to CHERI ISA
• CCall/CReturn hardware optimizations
• Linker support for capabilities
• CHERI multithreading/multicore
• Additional languages: Object C, Ocaml
Compartmentalized packet capture and processing

CHERI DEMO
November 2012 - CheriPoint

✓ Bespoke compartmentalized CHERI presentation package

✓ Sandboxing mitigates trojan inserted in PNG library

✗ Largely MIPS ISA code generated from C

✗ A small amount of utility code written in CHERI assembly

✗ Static sandboxing policy
CHERI tcpdump demonstration

- Memory protection + compartmentalization
  - OS support for CHERI thread contexts
  - Compiler `__capability` pointers
  - Userspace `libcheri` sandboxing model
  - Compartmentalized packet printing
- Key results:
  - Applicability of hybrid capability model
  - Tight C-language/capability integration
  - Tradeoffs policy/performance/mitigation
  - Compartmentalization scalability
  - Variable granularity
Software analysis and transformation

SOAAP AND TESLA
Security-oriented analysis of application programs (SOAAP)

- Static and dynamic analysis tools to assist programmers when compartmentalizing applications
- Come see demo at poster session!
TESLA

• Pragmatic validation of run-time security properties
• LTL-like assertions embedded in code
• Compiler-generated instrumentation
• Significant outreach to potential open-source and corporate consumers
• Come see demo at poster session!
TESLA since last time

- Applied TESLA to OpenSSL, FreeBSD, Objective-C
- Found subtle bugs that eluded traditional debug tools
- Build cost: rebuilds less incremental
- Significant runtime cost optimizations
Conclusion

- Three years into the five-year project
- Mature CHERI hardware platform
- CheriBSD operating system
- CHERI Clang/LLVM/LLDB/SDK
- CHERI application exploration in progress
- SOAAP and TESLA tools maturing
- Smten, architectural extraction, and formal ISA models bearing early verification results
(MRC)² sister project

- Heavy use of CTSRD-derived CHERI
- Multithreaded and multicore CHERI prototypes
- CHERI on NetFPGA 10G
Q&A
CTSRD at the PI Meeting

Dr Peter G. Neumann
Dr Robert N. M. Watson
Dr Simon W. Moore
Dr Jonathan Anderson
Dr David Chisnall
Dr Nirav Dave
Mr Brooks Davis
Mr Rance DeLong
Dr Khilan Gudka
Dr Theo A. Markettos
Mr Ed Maste
Dr Michael Roe
Mr Colin Rothwell
Mr Stacey Son